Apprentissage partiel de grammaires lexicalisées

Erwan Moreau

Apprentissage partiel de grammaires lexicalisées

Erwan Moreau

LINA - FRE CNRS 2729 - Université de Nantes
2 rue de la Houssinière - BP 92208 - 44322 Nantes cedex 3
Erwan.Moreau@univ-nantes.fr

RÉSUMÉ. *Sur le plan théorique, le modèle de Gold semble adapté à l'apprentissage des langues naturelles. Cependant la mise en pratique des algorithmes d'acquisition issus de ce modèle pose de nombreux problèmes. Nous développons dans cet article des résultats obtenus à la suite des travaux de Buszkowski, Penn et Kanazawa, qui ont montré que certaines classes de grammaires catégorielles sont apprenables. L'algorithme d'origine nécessite une grande quantité d'information en entrée pour être efficace. En changeant la nature des informations en entrée, nous proposons un algorithme d'apprentissage de grammaires catégorielles plus réaliste dans la perspective d'applications aux langues naturelles. Cette méthode peut être étendue à certains formalismes grammaticaux lexicalisés, comme les grammaires de liens. L'expérimentation que nous proposons avec ce formalisme tend à montrer la faisabilité de notre approche.*

ABSTRACT. *In a theoretical viewpoint, Gold's model suits well to natural language learning. But there are numerous problems in using learning algorithms resulting from this model in real applications. In particular, we develop in this article results obtained by Buszkowski, Penn and Kanazawa, who have shown that some classes of categorial grammars are learnable. The original algorithm is efficient but needs a lot of information as input. We propose a new algorithm which does not use the same kind of information as input, and is more realistic in the perspective of real applications to natural languages. It is possible to extend this method to different lexicalized grammatical formalisms, like link grammars. The experiments we propose in this framework tend to show the feasibility of our approach.*

MOTS-CLÉS : *Inférence grammaticale, apprentissage partiel, grammaires catégorielles, grammaires de liens.*

KEYWORDS: *Grammatical inference, partial learning, categorial grammars, link grammars.*

1re soumission à *TAL*, le 15 septembre 2005.

1. Introduction

Malgré la facilité presque surprenante avec laquelle un enfant est capable d'acquérir sa langue maternelle, la réalisation d'un tel processus par la machine est un problème très difficile. Ce problème de l'apprentissage automatique de grammaires consiste à découvrir les règles (syntaxiques) de formation des phrases d'un langage particulier. Plusieurs modèles de formalisation de ce processus existent. Le modèle de Gold [GOL 67] est celui que nous utilisons dans cet article, et plus spécifiquement dans ce modèle la méthode proposée par Buszkowski [BUS 89] et généralisée plus tard par Kanazawa [KAN 98].

Le modèle d'apprentissage proposé par Gold est très restrictif, c'est pourquoi les premiers résultats obtenus avec ce modèle ont été négatifs. Mais après les premiers résultats positifs obtenus par Angluin [ANG 80a], un certain nombre de travaux ont contribué à montrer la pertinence de ce modèle. Parmi ceux-ci, Kanazawa a montré que certaines classes de langages non triviales sont apprenables, en se servant de l'algorithme proposé par Buszkoswki pour les grammaires catégorielles classiques. Dans ce cadre, le mécanisme d'apprentissage est entièrement *symbolique*, c'est-à-dire à dire qu'il n'y a aucune approximation par rapport aux données fournies : en supposant qu'on dispose de données "parfaites" (ne comportant aucune erreur par rapport à la grammaire qu'elles sont censées représenter), le risque d'erreur est réduit à zéro. Mais cette précision a un prix : le bon fonctionnement de l'algorithme d'apprentissage doit satisfaire des contraintes très fortes.

L'une de ces contraintes qui font obstacle à l'application de cette méthode au langage naturel concerne la nature des données dont l'algorithme a besoin en entrée : il ne s'agit pas seulement de phrases simples mais de structures particulières, ce qui permet à l'algorithme d'être déterministe et efficace. Ces structures sont une forme "d'arbre de dérivation appauvri" des phrases considérées, dans le formalisme des grammaires catégorielles classiques. La disponibilité de telles structures dans des cas réels (i.e. pas seulement sur des exemples jouets) est loin d'être assurée, principalement pour deux raisons : d'une part le formalisme des grammaires catégorielles classiques est peu utilisé (en partie à cause de sa faible expressivité) ; d'autre part la connaissance de la grammaire sous-jacente est presque indispensable à la construction de ces structures, ce qui est un handicap majeur puisque l'objectif est précisément de déduire cette grammaire.

Nous proposons ici un compromis, basé sur la méthode de Kanazawa, qui conserve les avantages de l'apprentissage symbolique tout en éliminant cette contrainte sur les structures. En contrepartie, on considérera qu'une partie de la grammaire est déjà connue, de manière à remplacer l'information apportée par les structures par celle apportée par la *grammaire initiale*. Cette hypothèse est relativement réaliste dans la perspective de l'application aux cas réels, du fait notamment de la lexicalisation totale des grammaires catégorielles et de la "loi de Zipf" qui se vérifie sur les textes en langue naturelle. L'inconvénient étant que l'efficacité de l'apprentissage dépend désormais beaucoup de la grammaire initiale.

La première partie de cet article propose un bref survol des principales définitions et propriétés du modèle de Gold, de manière à donner au lecteur une perception plus claire des différences entre les algorithmes d'apprentissage de grammaires catégorielles présentés dans la seconde partie. Notre proposition d'algorithme d'apprentissage partiel est ensuite détaillée dans la troisième partie, puis soumise dans la dernière partie à une première expérimentation utilisant le lexique anglais des grammaires de liens.

2. Modèle de Gold : enjeux et limites

La principale difficulté de l'apprentissage à partir d'exemples positifs (c'est-à-dire dans le cas où l'on ne dispose que des éléments du langage à deviner, sans contre-exemples) est la *surgénéralisation*. La surgénéralisation est l'erreur qui consiste à trop généraliser (extrapoler) à partir des données fournies, comme l'indique Sakakibara dans [SAK 97] : *"Le problème est d'éviter la surgénéralisation, ce qui signifi e inférer un langage qui est un sur-ensemble strict du langage inconnu"*. Par exemple, on peut supposer que la séquence $3, 5, 7, x$ décrit l'ensemble des nombres impairs supérieurs à 2, et en déduire que x aura pour valeur 9. Mais s'il s'avère que x est 11, la séquence représentant en fait les nombres premiers strictement supérieurs à 2, alors il y a eu surgénéralisation : l'ensemble des nombres représenté est un sous-ensemble (strict) de l'ensemble qu'on a proposé [ANG 83]. Comme on ne dispose que d'exemples positifs, il n'y aura jamais de contre-exemple dans la séquence permettant de corriger l'erreur. Bien entendu, la généralisation est indispensable dans le processus d'apprentissage. En effet, une méthode d'apprentissage "trop prudente" qui ne généraliserait jamais ne ferait pas véritablement un *apprentissage*, au sens d'une découverte de quelque chose de nouveau : il s'agirait simplement d'une sorte de compilation des exemples proposés. Surtout, il est évident qu'une telle méthode serait incapable d'identifier un langage infini.

Sauf cas particuliers, la généralisation doit donc bien être utilisée au cours de l'apprentissage. La question qui se pose d'un point de vue algorithmique est : quand faut-il généraliser ? (ou quand faut-il ne pas généraliser, selon ce qu'on considère comme étant l'action par défaut). Mais avant de se poser cette question, il faut s'assurer qu'il est *possible de savoir quand généraliser*, car lorsqu'on ne dispose pas d'exemples négatifs on n'a aucun indice sur la position de la frontière du langage à deviner. C'est précisément ce point qui pose problème au départ avec l'identification à la limite à partir d'exemples positifs : Gold constate immédiatement après avoir défini son modèle que même les classes de langages qu'on croyait simples (les langages réguliers, voir exemple 1) ne sont pas apprenables, parce qu'on ne peut pas savoir quand [ne pas] généraliser.

La caractérisation de classes de langages apprenables passera donc d'abord par l'étude des cas où cette question peut être résolue. C'est seulement au début des années 1980 qu'Angluin apporte au modèle de Gold ses premiers résultats positifs :

dans [ANG 80b], elle propose de *"considérer le cas particulier d'inférence à partir d'exemples positifs qui évite la surgénéralisation [et donne] des conditions suffi santes pour cela"*. Le critère proposé par Angluin, basé sur les *ensembles révélateurs*, a donné lieu à de nombreuses utilisations ou extensions démontrant finalement la richesse du modèle de Gold. L'une des propriétés dérivée de ce premier résultat positif est *l'élasticité fi nie* proposée par Wright [WRI 89] [MOT 91], elle-même réutilisée avec succès par Shinohara dans la propriété de *densité fi nie bornée* [SHI 91].

2.1. *Identification à la limite : définition et conséquences directes*

Dans cette partie nous utiliserons le terme abstrait *objets* pour désigner les éléments d'un langage. Selon les cas, ces objets pourront désigner de simples séquences de mots (phrases), mais aussi des structures plus ou moins complexes. Les *ensembles* dont il est question ci-dessous peuvent être infinis, sauf précision contraire explicite. Le terme *apprenant* désigne ici un "mécanisme" dont la tâche est de fournir une représentation grammaticale d'un langage décrit seulement par ses objets.

Un *système de grammaires* est un triplet $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ dans lequel $\mathcal{U}$ est un ensemble d'objets, $\mathcal{G}$ un ensemble de grammaires et $\mathcal{M}$ une fonction qui associe à chaque grammaire de $\mathcal{G}$ un sous-ensemble de $\mathcal{U}$: $\mathcal{M}(G)$ est le *langage* représenté par la grammaire G. Dans un système de grammaires $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$, une *fonction d'apprentissage* associe à toute séquence finie d'objets de $\mathcal{U}$ une grammaire de $\mathcal{G}$.

Dans le modèle de Gold [GOL 67], une séquence infinie d'objets est présentée à l'apprenant, qui doit "deviner" le langage ainsi énuméré. À chaque nouvel exemple, l'apprenant doit proposer une hypothèse sous forme d'une grammaire. Il y a *convergence* si au bout d'un nombre fini d'exemples l'hypothèse de l'apprenant n'est plus modifiée. Si de plus cette dernière grammaire représente le langage correct, alors l'apprentissage est réalisé avec succès. Ce critère de succès est décrit formellement dans la définition suivante :

Définition 2.1 (Identification à la limite). *Soit $\langle \mathcal{U}, \mathcal{G}, \mathcal{M} \rangle$ un système de grammaires, ϕ une fonction d'apprentissage et $L \subseteq \mathcal{U}$ un langage. Soit $\langle a_1, a_2, \ldots \rangle$ une séquence infi nie d'objets de $\mathcal{U}$, telle que $a \in \{ a_i \mid i \in \mathbb{N} \}$ si et seulement si $a \in L$. ϕ converge vers G s'il existe $n \in \mathbb{N}$ tel que pour tout $i \geq n$ $\phi(\langle a_1, a_2, \ldots, a_i \rangle) = G$. Une classe de langages $\mathcal{L} \subseteq \mathcal{P}(\mathcal{U})$ est dite apprenable s'il existe une fonction d'apprentissage ϕ telle que, pour tout langage $L \in \mathcal{L}$ et toute énumération de ce langage, ϕ converge vers une grammaire G telle que $\mathcal{M}(G) = L$.*

Remarque : la convergence d'une fonction d'apprentissage n'a d'intérêt que si elle s'applique à un ensemble de langages, et non à un seul langage. En effet, plus la classe de langages est grande, plus la difficulté est importante de reconnaître un langage précis dans cette classe (en particulier, le cas d'un langage unique est trivial : il suffit à la fonction d'apprentissage de toujours renvoyer la même grammaire).

On voit dans cette définition que la séquence d'exemples a quelques caractéristiques notables :

– Elle ne contient que des exemples *positifs*. La fonction d'apprentissage n'a donc aucune information extérieure au langage, ce qui constitue une difficulté de taille pour situer les frontières du langage[1].

– Tous les objets du langage apparaissent dans la séquence (ceci est possible parce que la séquence est infinie).

– Les exemples peuvent apparaître dans un ordre quelconque dans la séquence (et éventuellement plusieurs fois). Le modèle n'impose pas que la fonction d'apprentissage donne le même résultat pour deux séquences différentes contenant un même ensemble d'exemples.

Cette définition de l'identification à la limite a d'importantes conséquences immédiates, démontrées par Gold dans [GOL 67]. La première est un résultat positif pour les langages finis : la classe des langages finis est apprenable, et a fortiori toute sous-classe ne contenant que des langages finis. En effet, il suffit dans ce cas que l'apprenant ajoute un par un les exemples présentés à la grammaire hypothèse. Comme le langage est fini et que tous ses éléments sont présents dans la séquence d'exemples, la fonction d'apprentissage converge dès que le langage a été énuméré en totalité. En revanche la seconde conséquence de la définition du modèle est un résultat négatif : toute classe de langages contenant tous les langages finis et au moins un langage infini n'est pas apprenable. Il s'agit en fait d'une conséquence du théorème plus général concernant l'existence d'un *point limite* (le terme étant dû à Kapur [KAP 91]) :

Théorème 2.2 (Point limite). *Un langage L_∞ est un point limite pour une classe de langages $\mathcal{L}$ s'il existe une séquence infinie $\langle L_1, L_2, \ldots \rangle$ de langages dans $\mathcal{L}$ telle que*

$$L_i \subset L_{i+1} \text{ pour tout } i > 0 \quad \text{et} \quad L_\infty = \bigcup_{n \in \mathbb{N}} L_n.$$

Si la classe de langages $\mathcal{L}$ a un point limite, alors $\mathcal{L}$ n'est pas apprenable.

On peut grossièrement résumer la preuve de ce théorème de la manière suivante : supposons qu'il existe une fonction d'aprentissage qui apprend $\mathcal{L}$ et qu'on lui présente des éléments du langage L_1, puis des éléments du langage L_2, et ainsi de suite à l'infini. A chaque étape n, la fonction d'apprentissage doit choisir entre le langage L_n et le langage L_∞ et ne dispose d'aucun indice pour cela : si elle choisit L_n (méthode prudente), elle n'atteindra jamais L_∞, donc ne converge pas vers le bon langage dans

1. En fait Gold a proposé plusieurs modèles d'apprentissage dans [GOL 67], dont celui "par informateur" dans lequel les exemples négatifs sont utilisés. Mais cette variante présente peu d'intérêt : d'une part parce qu'elle permet d'apprendre "trop de choses" (les langages primitifs récursifs), et d'autre part, d'un point de vue pratique, il est difficilement envisageable d'obtenir les données négatives complètes d'un langage. C'est pourquoi on assimile souvent le modèle de Gold seulement à sa variante qui n'utilise que les exemples positifs.

le cas d'une énumération de L_∞. A l'inverse, si à l'étape n la fonction renvoie comme hypothèse le langage L_∞ (généralisation), alors elle ne peut pas apprendre L_n. Par conséquent il est impossible d'apprendre une classe contenant un tel ensemble de langages.

Exemple 1. *Pour tout $n \geq 1$ on défi nit $L_n = \{ x^i \mid i \leq n \}$ comme le langage des chaînes de x de longueur inférieure ou égale à n. Soit $L_\infty = x^*$ le langage de toutes les chaînes de x, et $\mathcal{L}$ une classe de langages contenant tous les L_n ($n \in \mathbb{N}$) ainsi que L_∞.*

On voit que $L_{n+1} = L_n \cup \{x^{n+1}\}$ pour tout n, donc $L_n \subset L_{n+1}$. De plus tout langage L_n est inclus (strictement) dans L_∞, qui est par conséquent un point limite pour $\mathcal{L}$. Donc $\mathcal{L}$ n'est pas apprenable.

Les langages $\{ L_n \mid n \in \mathbb{N} \}$ et L_∞ définis dans l'exemple ci-dessus sont tous réguliers, ce qui amène à un corollaire important du théorème 2.2 : ces langages étant réguliers, toutes les classes de la hiérarchie de Chomsky les contiennent, et ne sont par conséquent pas apprenables. Le fait que même la classe la plus simple, celle des langages réguliers, ne soit pas apprenable dans le modèle de Gold a longtemps constitué un obstacle majeur au développement du modèle, car il était considéré comme "trop contraignant".

2.2. *Principaux résultats d'apprenabilité*

Les premiers résultats positifs pour des classes de langages non triviales dans le modèle de Gold sont dûs à Angluin au début des années 80 [ANG 80a] [ANG 80b]. Ils sont suivis par un certain nombre de travaux concernant principalement le modèle d'identification à la limite lui-même, c'est-à-dire la caractérisation de ce qui est apprenable et de ce qui ne l'est pas. En particulier, le critère d'*élasticité fi nie* (défini une première fois par Wright dans [WRI 89] puis corrigé dans [MOT 91]) est une condition suffisante très utile pour démontrer l'apprenabilité.

Théorème 2.3 (Élasticité finie [WRI 89]). *Une classe de langages $\mathcal{L}$ a la propriété d'élasticité infi nie s'il existe une séquence infi nie d'objets $\langle a_0, a_1, \ldots \rangle$ et une séquence infi nie de langages $\langle L_1, L_2, \ldots \rangle$ tels que pour tout $n \geq 1$:*

$$\{a_0, a_1, \ldots, a_{n-1}\} \subseteq L_n \text{ , mais } a_n \notin L_n \text{ .}$$

Une classe de langages a la propriété d'élasticité fi nie si elle n'a pas l'élasticité infi nie.

Ce critère est utilisé notamment par Shinohara [SHI 91] pour démontrer l'apprenabilité de la sous-classe des langages contextuels $\mathcal{L}_{k-CSG}$, constituée des langages définis par des grammaires d'au plus k règles, k étant un entier fixé[2].

2. Il est important de noter que même si k, le nombre maximum de règles des grammaires, peut prendre n'importe quelle valeur, cette valeur est *fixée a priori* : cela signifi e qu'on ne peut pas

En quoi est-ce un résultat d'apprenabilité "non trivial" ? Il ne s'agit pas ici de l'importante expressivité de la classe de langages ni de la complexité de la preuve de Shinohara, mais du fait que la classe de langages est complexe à apprendre : elle contient un nombre infini de langages, finis et infinis, car la borne k sur le nombre de règles ne limite pas la longueur des règles. Néanmoins le fait qu'un nombre infini de langages finis et de langages infinis appartiennent à cette classe ne contredit pas le théorème de Gold, parce que la classe $\mathcal{L}_{k-CSG}$ ne contient pas *tous* les langages finis.

Exemple 2. *Soit G_n la grammaire régulière définie par l'ensemble de règles $\{ S \rightarrow a^n\, b\, S \ ; \ S \rightarrow c \}$. Pour tout $n > 0$, le langage défini par G_n est $\{ (a^n\, b)^*\, c \}$: le langage de G_n est donc toujours infini. De plus, la chaîne $a^p\, b\, c$ appartient au langage de G_m si et seulement si $m = n$, donc l'ensemble des langages correspondant aux grammaires de l'ensemble $\{ G_n \mid n > 0 \}$ est infini.*

Cette classe de langages contient par conséquent un nombre infini de langages infinis, définis par des grammaires ne contenant que deux règles. En revanche, il est impossible de construire l'ensemble infini des chaînes de "a" de longueur inférieure ou égale à n (comme dans l'exemple 1) avec un nombre de règles inférieur à un k fixé.

Dans [KAN 98], Kanazawa démontre que la classe des langages de chaînes des grammaires AB k-valuées possède l'élasticité finie (et est donc apprenable). Il démontre pour cela la propriété suivante sur l'élasticité finie :

Théorème 2.4 (Élasticité finie et relation finiment valuée [KAN 98]). *Une relation $R \subseteq A \times B$ est finiment valuée si et seulement si pour tout $a \in A$ il n'existe au plus qu'un nombre fini de $b \in B$ tels que $a\ R\ b$.*

*Si L est un langage sur $\mathcal{U}$ et R une relation sur $\mathcal{U}' \times \mathcal{U}$, l'*image inverse *de L par rapport à R est le langage $R^{-1}[L] = \{ a \in \mathcal{U}' \mid \exists b \in L \text{ tel que } a\ R\ b \}$.*

Soient $\mathcal{U}$ et $\mathcal{U}'$ deux ensembles d'objets, et $\mathcal{L}$ une classe de langages définie sur $\mathcal{U}$ qui a l'élasticité finie.

S'il existe une relation $R \subseteq \mathcal{U}' \times \mathcal{U}$ finiment valuée, alors la classe de langages $\mathcal{L}' = \{ R^{-1}[L] \mid L \in \mathcal{L} \}$ a aussi l'élasticité finie.

Ce théorème s'avère très utile pour démontrer l'élasticité finie d'une classe dans le cas où il existe une classe "voisine" pour laquelle l'élasticité finie est déjà prouvée. Il est ainsi possible d'étendre l'élasticité finie à une classe plus complexe, dès lors qu'une relation finiment valuée peut être établie entre les univers de définition des deux classes. Cette méthode est comparable à la réduction d'un problème à un autre, utilisée par exemple pour montrer la NP-complétude du problème. Certaines conséquences de l'utilisation de cette technique pour l'apprentissage des grammaires catégorielles seront détaillées dans la partie 3.3.

---

ajouter des règles à l'infini (et donc qu'il est impossible de construire un point limite, ce qui serait une contradiction).

2.3. *Application(s) aux langues naturelles ?*

Le modèle de Gold est un cadre formel possible du processus de l'apprentissage. Mais est-il pertinent de l'utiliser pour représenter l'acquisition des langues naturelles ? Tout d'abord, ce modèle théorique semble répondre relativement bien à ce qu'on en attend intuitivement : l'apprenant est face à une séquence infinie d'exemples du langage, et fait évoluer sa conception de ce langage en fonction des exemples vus. L'apprenant n'est jamais informé du fait qu'il a atteint le langage-cible, mais le modèle garantit (dans le cas d'une classe apprenable) qu'il est possible de l'atteindre avec un nombre fini d'exemples. De plus, certaines caractéristiques du modèle sont en accord avec les observations linguistiques sur l'acquistion du langage par l'enfant : l'enfant apprend sa langue maternelle en utilisant uniquement les exemples positifs dont il dispose[3] (les phrases du langage prononcées par son entourage). Enfin le point positif le plus important est sans doute le résultat de Shinohara [SHI 91] sur l'apprenabilité des langages définis par des grammaires contextuelles d'au plus k règles : si l'on suppose qu'il est possible de représenter tout langage naturel sous forme d'une grammaire contextuelle finie, alors selon ce résultat la classe des langues naturelles est apprenable dans le modèle de Gold, puisqu'il existe un nombre k suffisamment grand pout représenter l'ensemble des langues naturelles sous forme de grammaires contextuelles.

Néanmoins, il ne faut pas prêter à ce modèle plus de qualités qu'il n'en a. Par exemple, l'infinité de la séquence d'exemples positifs ne doit pas être confondue avec l'aspect dynamique des langues naturelles (le fait qu'une langue évolue au cours du temps, au niveau du vocabulaire ou de la grammaire) : dans le modèle de Gold, le postulat de base est que le langage existe dans la classe étudiée, et est par conséquent fixé, même si on ne sait pas encore le représenter, dès le premier exemple. En particulier, il est possible qu'un mot n'apparaisse que "très tard" dans la séquence de phrases, mais l'ensemble du vocabulaire est fixé au départ (sans quoi la notion de convergence n'aurait aucun sens, puisque si on peut toujours ajouter un mot au vocabulaire, donc créer une nouvelle phrase du langage, on voit bien qu'aucun algorithme ne peut converger vers ce langage). De plus, la mise en pratique de l'apprentissage dans le modèle de Gold se différencie souvent (notamment dans le cas des algorithmes de type Buszkoswki/Kanazawa) de l'acquisition humaine par le fait que ces modèles formels font croître le langage jusqu'au langage cible, tandis que l'acquisition naturelle fait au contraire décroître le langage, du plus général au plus spécifique. Enfin le modèle reste à un niveau d'abstration très élevé pour la représentation des langages, qui est un point important pour les langues naturelles. On peut en effet noter que les éléments de base dans cette définition de l'apprenabilité sont les *objets* du langage, compris comme étant de simples éléments d'un ensemble. Or dans le cas d'un langage naturel, les objets sont des phrases (éventuellement enrichies d'informations sur leur structure), ce qui signifie que le modèle n'impose aucune contrainte sur (par exemple) la décompo-

3. Il est aujourd'hui communément admis dans la communauté linguistique que les enfants acquièrent le langage quasiment uniquement à partir d'exemples positifs (voir par exemple [PIN 94], [MAR 93]). En particulier l'enfant utilise très peu les éventuelles corrections indiquées par son entourage.

sition en mots ou la sémantique. Dans un sens il s'agit bien sûr d'un avantage, puisque cela laisse toute liberté sur le choix du formalisme utilisé, mais cela montre aussi que le modèle n'est pas spécifiquement conçu pour les langues naturelles.

On peut distinguer deux directions relativement distinctes dans l'application du modèle de Gold aux langues naturelles. La première direction travaille sur l'hypothèse que ce modèle est une représentation théorique fiable du processus d'acquisition du langage par l'enfant. Dans ce cadre, l'objectif consiste principalement à valider ou invalider des hypothèses sur le mode d'apprentissage de l'enfant, notamment en simulant cette acquisition par la machine. Par exemple, Dudau-Sofronie et Tellier [DUD 04b] [DUD 04a] adaptent les résultats de Kanazawa sur les grammaires catégorielles de manière à prendre en compte des informations d'ordre sémantique dans les exemples, conformément à ce dont dispose un enfant dans son environnement. La seconde direction, à laquelle se rattache le présent article, vise à mettre en pratique l'apprentissage par la machine, dans l'objectif d'en tirer des informations (idéalement, une grammaire précise et exhaustive) potentiellement utilisables pour d'autres tâches. Ce passage de la théorie à la pratique pose les problèmes complexes de l'existence / l'obtention de données correctes comme séquence d'exemples (avec pour corollaire la question de la nature des données - voir la partie 4.1) et évidemment de l'efficacité des algorithmes.

Dans les deux cas, il faut que l'apprenabilité de la classe de langages considérée soit démontrée avant d'envisager la construction d'un algorithme d'apprentissage : vouloir établir un tel algorithme sans passer par cette première étape serait en quelque sorte comme tenter de réaliser un algorithme pour résoudre un problème dont on ignore s'il est décidable. Cette recherche de l'apprenabilité est fortement liée au formalisme grammatical utilisé, puisqu'il est souvent nécessaire de restreindre la classe de langages en fonction de critères propres à ce formalisme (comme dans le cas des grammaires contextuelles dont le nombre de règles est borné). Mais si la preuve d'apprenabilité formelle induit souvent un algorithme convergent, elle ne garantit pas en revanche l'efficacité de celui-ci[4]. Le passage à l'échelle est donc certainement le principal obstacle à l'utilisation du modèle de Gold en traitement automatique des langues, puisque la plupart des algorithmes "intéressants" dans ce cadre sont de complexité exponentielle[5].

4. L'apprenabilité n'est d'ailleurs même pas une condition suffi sante à la simple existence d'un algorithme d'apprentissage.

5. Il faut souligner que la plupart des algorithmes d'apprentissage fonctionnent en *construisant* une grammaire du langage présenté, et non en *choisissant* parmi un ensemble de grammaires celle qui correspond au langage. Cette dernière solution ne contredit pas le modèle de Gold, mais n'est utilisable en pratique que dans le cas d'un ensemble fi ni et très restreint de langages bien identifi és (ce qui correspondrait plutôt en TALN au problème de la détection automatique de la langue).

3. Apprentissage de grammaires catégorielles

3.1. *Grammaires AB*

Les grammaires catégorielles classiques, nommées aussi grammaires AB, ont été introduites dans [BAR 53]. Ces grammaires sont totalement lexicalisées : cela signifie qu'une grammaire est décrite uniquement par son lexique, le lexique étant l'association d'une ou plusieurs catégories à chaque mot du vocabulaire. Les règles utilisées dans les dérivations sont donc *universelles*. Ces règles sont :

$$A/B,\ B \rightarrow A \quad \textit{FA (Forward Application)}$$
$$B,\ B\backslash A \rightarrow A \quad \textit{BA (Backward Application)}$$

Les *catégories* (aussi nommées *types*) sont des termes utilisant les opérateurs binaires $/$ et $\backslash$. Intuitivement, une expression est de type A/B (resp. $B \setminus A$) si cette expression est de type A lorsqu'elle est suivie (resp. précédée) par une expression de type B. Une phrase est correcte s'il est possible d'associer à chaque mot l'une de ses catégories, de telle sorte que les règles universelles permettent de transformer cette séquence de catégories en la catégorie spéciale S.

Exemple 3. *Soit G la grammaire constituée du lexique suivant :*

$\{$*Pierre, Marie, Paul* : SN; *aime, déteste* : $(SN\backslash S)/SN$; *qui* : $(SN\backslash SN)/(SN\backslash S)\}$.

La phrase "Pierre, qui aime Marie, déteste Paul" appartient au langage de cette grammaire, comme le montre la dérivation suivante :

$$\begin{array}{ll} & SN,\ (SN\backslash SN)/(SN\backslash S),\ (SN\backslash S)/SN,\ SN,\ (SN\backslash S)/SN,\ SN \\ \Rightarrow & SN,\ (SN\backslash SN)/(SN\backslash S),\ (SN\backslash S)/SN,\ SN,\ SN\backslash S \\ \Rightarrow & SN,\ (SN\backslash SN)/(SN\backslash S),\ SN\backslash S,\ SN\backslash S \\ \Rightarrow & SN,\ SN\backslash SN,\ SN\backslash S \\ \Rightarrow & SN,\ SN\backslash S \\ \Rightarrow & S \end{array}$$

Une grammaire AB est *rigide* si chaque mot n'est défini que par une seul catégorie. De même, une grammaire est dite *k-valuée* si chaque mot est défini par au plus k catégories.

On peut décrire une dérivation à l'aide d'un arbre de manière classique, en étiquetant les nœuds par la catégorie du constituant qu'ils représentent. De plus, la forme des règles permet aussi de représenter un arbre de dérivation en étiquetant les nœuds seulement par l'identifiant de la règle utilisée (FA ou BA). Une telle structure, dans laquelle les feuilles ne sont étiquetées que par un mot, est appellée *FA-structure* (pour *Functor-Argument structure*). Cette représention est unique pour un arbre de dérivation donné (voir figure 1). En revanche une FA-structure donnée peut représenter un nombre infini d'arbres de dérivations : dans la figure 1, on peut par exemple remplacer tous les SN par des $(X_1/X_2)/.../X_n$ et la FA-structure reste identique.

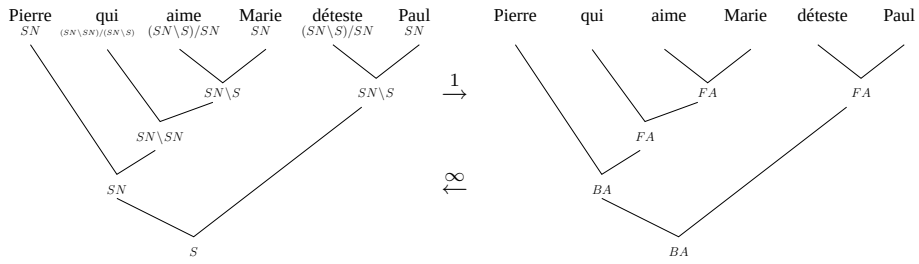


Figure 1. *Arbres de dérivations et FA-structure*

Cette notion de FA-structure est à la base de l'algorithme d'apprentissage de grammaires AB défini par Buszkowski [BUS 89] puis enrichi par Kanazawa [KAN 98]. Dans la mesure où ces structures se résument à un parenthésage en constituants enrichi d'une information d'orientation (qu'on peut assimiler au sens d'une dépendance entre constituants), elles sont un intermédiaire assez équilibré entre les simples chaînes (phrases) du langage et leurs arbres de dérivation complets. L'ajout de ces informations structurelles pour l'apprentissage est généralement justifié par le fait que l'enfant acquérant un langage dispose dans son environnement "d'informations complémentaires" qui l'aident dans son apprentissage. Étant donné l'extrême précision des FA-structures par rapport à la grammaire, il est cependant discutable de considérer que l'enfant a à sa disposition de telles informations. Ci-dessous sont définies plus formellement les notions de FA-structures et de langages de structures / de chaînes :

Définition 3.1 (FA-structure). *Étant donné un vocabulaire Σ, l'ensemble des FA-structures $\mathcal{SL}_\Sigma$ est défi ni comme le plus petit ensemble tel que*

– Tout mot $w \in \Sigma$ appartient à $\mathcal{SL}_\Sigma$,

– Pour tout couple $T_1, T_2 \in \mathcal{SL}_\Sigma$, les FA-structures $FA(T_1, T_2)$ et $BA(T_1, T_2)$ appartiennent à $\mathcal{SL}_\Sigma$.

Le *produit* d'une FA-structure est défini comme la séquence des mots qui apparaissent aux feuilles de celle-ci, c'est-à-dire la phrase représentée par cette structure.

Définition 3.2 (Instance d'une FA-structure). *Soit G une grammaire AB et T une FA-structure. Un couple (α, t_0), où α est une séquence non vide de catégories et t_0 une catégorie, est une instance de T pour G si l'une des conditions suivantes est remplie :*

– si T est constituée uniquement d'un mot w, alors $\alpha = \langle t_0 \rangle$ et il existe une règle de G qui associe la catégorie t_0 à w ($w \mapsto t_0$).

– si $T = FA(T_1, T_2)$ (resp. $T = BA(T_1, T_2)$), alors il existe deux couples (α_1, t_1) et (α_2, t_2), respectivement instances de T_1 et T_2, tels que $t_1 t_2 \rightarrow_{FA} t_0$ (resp. $t_1 t_2 \rightarrow_{BA} t_0$) et $\alpha = \alpha_1 \bullet \alpha_2$.

Intuitivement un couple (α, t) est une instance d'une FA-structure T pour une grammaire donnée si le fait de "coller" la séquence de types α aux feuilles de la struc-

ture et le type t à la racine (et d'étiqueter les autres nœuds en conséquence) permet d'obtenir un arbre de dérivation partiel correct pour la grammaire. On peut ainsi définir le *langage de structures* d'une grammaire G, noté $SL(G)$, comme l'ensemble des FA-structures telles qu'il existe une instance (α, S) de T pour G. Le *langage de chaînes* de la grammaire G est alors défini comme l'ensemble des chaînes x telles qu'il existe une structure $T \in SL(G)$ dont x est le produit.

3.2. *L'algorithme RG*

L'algorithme RG (pour *Rigid Grammars*), proposé par Buszkowski [BUS 89], apprend la classe des grammaires AB rigides à partir de FA-structures (au sens du modèle de Gold).

3.2.1. *Algorithme*

L'algorithme RG comporte deux étapes. La première consiste à construire une *grammaire générale* à partir de la séquence de FA-structures fournies comme exemples :

Soit $D = \langle T_1, .., T_n \rangle$ la séquence de FA-structures en entrée.

1) Etiquetage d'une structure T_i :

a) la racine de T_i est étiquetée par le type S (le type primitif qui caractérise les phrases correctes).

b) en allant de la racine vers les feuilles, les fils de chaque nœud t sont étiquetés de la manière suivante : une nouvelle variable x est créée, avec laquelle le nœud argument est étiqueté. L'autre nœud est étiqueté t/x ou $x \setminus t$ selon qu'il s'agit d'un nœud FA ou BA. Le nœud argument est celui de la branche droite s'il s'agit d'un nœud FA, gauche si c'est un nœud BA.

2) Soit $\langle P_1..P_n \rangle$ l'ensemble des arbres de dérivations construits par étiquetage des FA-structures $\langle T_1..T_n \rangle$. Pour chaque arbre P_i et chaque feuille de cet arbre, une règle $w \mapsto t$ est créée, où w est le mot correspondant à cette feuille et t le type obtenu après étiquetage pour cette feuille. La grammaire ainsi obtenue est la grammaire générale $GF(D)$.

De fait, la phase d'étiquetage se résume à rassembler toutes les contraintes données par les exemples dans une grammaire unique. Par construction, chaque FA-structure de l'ensemble donné en entrée appartient au langage de structures de cette grammaire (et par conséquent le produit de la structure appartient au langage de chaînes de la grammaire).

Mais comme chaque occurence d'un mot dans un exemple se traduit par l'ajout d'une nouvelle règle, il est évident que cette seule étape ne permet pas d'apprendre le langage décrit puisque la taille de la grammaire augmente indéfiniment. C'est la raison pour laquelle la seconde étape *d'unifi cation* doit transformer $GF(D)$ en une grammaire rigide :

1) Pour chaque mot w défini dans $GF(D)$, soit $\mathcal{A}_w$ l'ensemble des types associés au mot w. Soit $\mathcal{A}$ l'union de tous les $\mathcal{A}_w$. Soit σ_u *l'unifieur le plus général* (MGU) de $\mathcal{A}$: un unifieur de $\mathcal{A}$ est une substitution σ telle que pour tout couple de types (t_1, t_2) de tout ensemble $\mathcal{A}_w$ on a $\sigma(t_1) = \sigma(t_2)$. Un unifieur σ_u est le plus général si pour tout autre unifieur σ il existe une substition τ qui permet de passer de σ_u à σ, i.e. $\sigma = \tau \circ \sigma_u$ (voir par exemple [KNI 89] pour des détails sur les algorithmes d'unification).

2) On définit la grammaire $RG(D)$ par $RG(D) = \sigma_u[GF(D)]$: toute règle $w \mapsto t$ de $GF(D)$ est remplacée par $w \mapsto \sigma_u(t)$ dans $RG(D)$. Comme tous les types d'un même mot ont été unifiés, la grammaire obtenue est rigide (par exemple, si $GF(D)$ contenait des règles $w \mapsto t_1$ et $w \mapsto t_2$, par définition de σ_u on a $\sigma_u(t_1) = \sigma_u(t_2)$).

Kanazawa donne dans [KAN 98] une preuve de convergence pour cet algorithme, démontrant ainsi que les langages de FA-structures de grammaires AB rigides sont apprenables (ce qu'on simplifie dans la littérature du domaine en "les grammaires AB rigides sont apprenables à partir de structures").

Cet algorithme a quelques qualités intéressantes : il est tout d'abord efficace, puisque sa complexité algorithmique (en fonction de la taille des exemples) est seulement quadratique[6]. Il peut être utilisé de manière incrémentale, ainsi il n'est pas nécessaire de recalculer la grammaire à partir de l'ensemble des phrases à chaque nouvel exemple. Enfin il produit une unique grammaire solution (la plus générale) quel que soit l'ensemble d'exemples proposés.

3.2.2. *Exemple*

On considère l'ensemble D de FA-structures représentées (après étiquetage des nœuds) sur la figure 2. La phase d'étiquetage permet d'obtenir la grammaire générale $GF(D)$ suivante :

6. Il ne faut pas confondre la complexité de l'algorithme d'apprentissage et la complexité de l'apprentissage en lui-même : la complexité de l'algorithme est le nombre de calculs élémentaires (au sens classique, c'est-à-dire dans une représentation raisonnable telle qu'une machine de Turing) nécessaire pour calculer la grammaire, en fonction de la taille de l'ensemble d'exemples (on choisit un critère de taille raisonnable, typiquement la somme du nombre de mots de chaque phrase). Mais la grammaire ainsi calculée n'est pas nécessairement la grammaire cible, puisque le modèle ne donne aucun moyen de prédire le nombre d'exemples nécessaires pour l'atteindre. Par conséquent on peut imaginer des cas où il faut un nombre exponentiel d'exemples (en fonction de la taille du vocabulaire par exemple) pour réaliser l'apprentissage : la complexité du processus global d'apprentissage n'est pas calculable a priori (sauf cas particuliers).

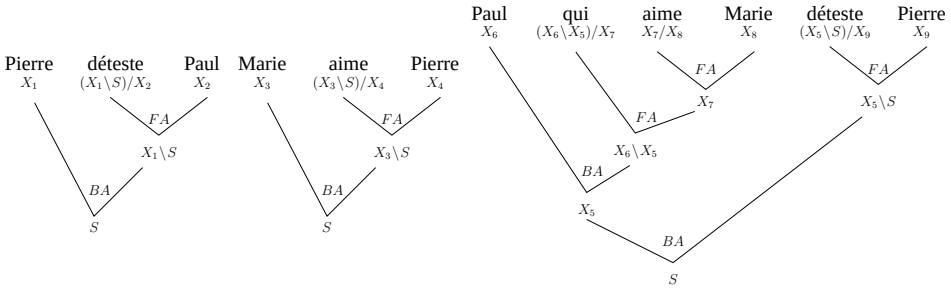


Figure 2. *FA-structures après étiquetage*

$$GF(D) = \begin{cases} \text{Pierre} & \mapsto & X_1, X_4, X_9 \\ \text{Marie} & \mapsto & X_3, X_8 \\ \text{Paul} & \mapsto & X_2, X_6 \\ \text{aime} & \mapsto & (X_3\backslash S)/X_4, X_7/X_8 \\ \text{déteste} & \mapsto & (X_1\backslash S)/X_2, (X_5\backslash S)/X_9 \\ \text{qui} & \mapsto & (X_6\backslash X_5)/X_7 \end{cases}$$

Le calcul du MGU unifie les types suivants : $\begin{cases} X_1 = X_4 = X_9, \\ X_3 = X_8, \\ X_2 = X_6, \\ X_7 = (X_3\backslash S), \\ X_4 = X_8, \\ X_1 = X_5, \\ X_2 = X_9 \end{cases}$

Ce qui donne : $X_1 = X_2 = X_3 = X_4 = X_5 = X_6 = X_8 = X_9$ et $X_7 = X_1\backslash S$. Donc la grammaire $RG(D)$ est :

$$RG(D) = \begin{cases} \text{Pierre} & \mapsto & X_1 \\ \text{Marie} & \mapsto & X_1 \\ \text{Paul} & \mapsto & X_1 \\ \text{aime} & \mapsto & (X_1\backslash S)/X_1 \\ \text{déteste} & \mapsto & (X_1\backslash S)/X_1 \\ \text{qui} & \mapsto & (X_1\backslash X_1)/(X_1\backslash S) \end{cases}$$

3.3. *Apprentissage de grammaires non rigides à partir de phrases plates*

L'apprentissage *à partir de phrases plates (flat strings)* désigne l'apprentissage des langages de chaînes, par opposition à l'apprentissage à partir de structures. La classe

de langages des grammaires AB est équivalente à celle des grammaires hors-contexte [BAR 60], elle n'est donc pas apprenable en totalité selon le théorème 2.2.

Du point de vue de l'application aux langues naturelles, l'algorithme RG présente peu d'intérêt, d'une part à cause de la nature des informations qu'il nécessite, et d'autre part à cause de la faible expressivité des grammaires rigides :

3.3.1. *Des FA-structures aux phrases plates*

Les exemples en entrée sont des FA-structures, information assez précise et surtout très liée au formalisme des grammaires catégorielles classiques. En conséquence il semble très difficile d'obtenir de telles données en grande quantité pour des langues naturelles. Kanazawa démontre cependant dans [KAN 98] que les grammaires AB rigides sont apprenables à partir de phrases plates. La classe des langages de FA-structures rigides ayant l'élasticité finie, il utilise pour cela le théorème 2.4 (qu'il a démontré) de la relation finiment valuée entre classes de langages. En effet, étant donné une phrase de taille n, il n'existe qu'un nombre fini de FA-structures qui peuvent lui correspondre : il y a "seulement" $\frac{(2n+2)!}{(n+1)!(n+2)!}$ parenthésages possibles (nombres de Catalan), et le nombre d'étiquetages possibles des nœuds par FA/BA est de 2^{n-1}. Ainsi la classe des langages de phrases plates rigides a aussi l'élasticité finie, et est par conséquent apprenable. L'algorithme proposé par Kanazawa consiste simplement à calculer toutes les grammaires possibles en fonction de toutes les FA-structures possibles, ce qui est bien sûr de complexité exponentielle. Costa-Florêncio montrera ensuite dans [COS 02] qu'il s'agit d'un problème NP-difficile, ce qui signifie que l'agorithme exponentiel de Kanazawa est difficilement améliorable.

3.3.2. *Des grammaires rigides aux grammaires k-valuées*

L'algorithme RG n'apprend que la classe des grammaires AB rigides, c'est-à-dire celle où chaque mot n'est défini que par une seule catégorie. C'est un manque d'expressivité rédhibitoire pour les langues naturelles, puisque cela exclut non seulement toutes les possibilités d'homonymie, mais surtout cela restreint considérablement l'usage de mots grammaticaux, qui doivent souvent pouvoir être utilisés dans plusieurs formes grammaticales différentes. Par exemple, le mot "le" est à la fois un déterminant et un pronom personnel, qui correspondent nécessairement à deux catégories syntaxiques différentes[7].

7. La difficulté consiste à éviter la *surgénéralisation*. En effet, rien n'interdit d'unifier deux types d'un même mot (par exemple ceux correspondant au mot "marche" en tant que verbe et nom commun), de manière à rendre la grammaire rigide tout en permettant à ce mot d'être utilisé avec les deux fonctions syntaxiques. Mais si tous les types de chaque mot sont unifiés de la sorte, la propagation des substitutions va entraîner des erreurs, sous forme de l'acceptation par la grammaire de phrases non correctes (par exemple "marche" et "mange" sont tous les deux des verbes, donc leurs types seront unifiés, ce qui permettra d'utiliser "mange" en tant que nom commun : erreur).

Kanazawa démontre aussi dans ce cas qu'il est possible d'apprendre la classe plus large des grammaires k-valuées, k étant un entier fixé. La méthode qu'il utilise est identique au cas précédent, à savoir établir une relation finiment valuée entre les grammaires rigides et les grammaires k-valuées (les deux classes ayant donc l'élasticité finie). Mais le problème de l'efficacité se pose encore dans ce cas du fait que l'algorithme, qui consiste à calculer tous les unifieurs k-*partiels* au lieu d'un unique MGU, est exponentiel (Costa-Florêncio montre aussi dans ce cas que le problème est NP-difficile [COS 01]).

Finalement dans ces deux cas l'extension à des classes de langages plus riches mène inévitablement à un problème d'efficacité, quasiment insoluble pour des applications réelles. Ceci illustre le fait que, même si l'élasticité finie est un critère très puissant pour démontrer l'apprenabilité "théorique" d'une classe de langages complexe à partir d'un cas simple, son utilisation ne préserve pas les propriétés algorithmiques du cas de base. Il est utile de noter que le problème des structures est très lié à la première partie de l'algorithme (construction de la grammaire générale), tandis que le problème de la rigidité des grammaires est lié à la seconde partie (l'unification). En fait il s'avère impossible de réaliser la première partie de l'algorithme en temps polynomial sans les FA-structures complètes, et de la même manière le déterminisme de la seconde partie est basé sur la rigidité des grammaires.

Bien entendu les deux résultats de Kanazawa peuvent être additionnés, donc la classe des grammaires k-valuées est apprenable à partir de phrases plates dans le modèle de Gold. Sur le plan théorique il s'agit d'un résultat relativement intéressant pour les langues naturelles, mais la mise en pratique est évidemment impossible. Nicolas a implémenté et testé l'algorithme de Kanazawa pour ce cas, et obtient par exemple 126775 grammaires solutions en fixant seulement la valeur 2 à k, pour de petits exemples [NIC 99]. Le fait que l'algorithme ne soit plus déterministe ajoute encore la complication de choisir une grammaire parmi toutes les solutions, de manière à satisfaire la définition du modèle de Gold (une seule grammaire en sortie). Kanazawa décrit les différents aspects de ce problème de la recherche d'une grammaire minimale dans l'ensemble de solutions, qui passe encore une fois par un algorithme exponentiel.

4. Apprentissage partiel de grammaires rigides

L'application des algorithmes d'apprentissage de grammaires catégorielles rigides se heurte donc au problème classique du rapport entre quantité d'information en entrée et efficacité de l'algorithme : soit l'algorithme est polynomial mais nécessite des FA-structures trop complexes, soit l'algorithme n'utilise que des phrases plates mais est alors exponentiel, donc tout aussi peu utilisable dans des applications réelles. Précisons que l'on s'intéresse ici uniquement au problème de l'apprentissage sans FA-structures, en restant dans le cas des grammaires rigides.

4.1. *Méthode*

Nous proposons ici un compromis dans lequel les informations que constituent les FA-structures sont remplacées par celles fournies par une *grammaire initiale*, de manière à maintenir un niveau acceptable d'efficacité. Comme les grammaires catégorielles sont totalement lexicalisées, la forme de cette grammaire initiale est simple : il s'agit d'un sous-ensemble (quelconque a priori) de la grammaire à apprendre, autrement dit d'un sous-ensemble de l'ensemble des règles lexicales de cette grammaire. Comme chaque règle est l'association d'un ou plusieurs types à un mot, on peut facilement interpréter ce sous-ensemble comme une grammaire incomplète, dans laquelle le ou les types de certains mots sont au départ inconnus : le but de l'apprentissage demeure bien sûr de trouver les types de ces mots. De plus, on verra dans la partie 4.4.2 que les textes en langage naturel ont certaines propriétés particulières (la "loi de Zipf") qui permettent d'utiliser au mieux cette méthode d'apprentissage partiel.

Afin de montrer combien l'analyse syntaxique et l'apprentissage sont liés dans le cas des grammaires catégorielles, nous proposons ci-dessous un petit programme Prolog qui est à la fois un analyseur syntaxique de grammaires AB, un algorithme d'apprentissage de grammaires rigides et un algorithme d'apprentissage partiel. Le point commun entre l'analyse et l'acquisition est la recherche de la FA-structure : dans le premier cas celle-ci garantit l'appartenance de la phrase au langage en utilisant les catégories, et dans le second elle induit les catégories des mots inconnus. Cet algorithme naïf, qui repose entièrement sur le moteur Prolog pour la recherche d'une dérivation et/ou le calcul des types inconnus du lexique, illustre aussi l'importance de l'unification dans le processus de dérivation des grammaires catégorielles.

```
:-op(400,xfx,/).
:-op(400,xfx,\).

regle(A/B,B,A).   % Forward Application (FA)
regle(B,B\A,A).   % Backward Application (BA)


couper_liste(Part1, Part2, Liste) :-
        append(Part1, Part2, Liste),
        Part1 \= [],
        Part2 \= [].

deriv(Lexique, [Mot], T) :-
        member(def(Mot,T), Lexique).
deriv(Lexique, Constituant, T) :-
        couper_liste(C1, C2, Constituant),
        deriv(Lexique, C1, T1),
        deriv(Lexique, C2, T2),
        regle(T1,T2,T).
```

```
exemple(X) :-    % 'Marie' de type inconnu :
  Lex = [        %  renvoie X = sn
        def('Pierre', sn),
        def('aime', (sn\s)/sn),
        def('Marie', X)
        ],
  deriv(Lex, [ 'Pierre', 'aime', 'Marie' ], s),
  deriv(Lex, [ 'Marie', 'aime', 'Pierre' ], s).
```

On peut noter que cet algorithme termine toujours, car le nombre de parenthésages d'une phrase en constituants (non vides) est borné. Cependant il est très peu efficace, surtout dans le cas où un grand nombre de mots sont définis dans le lexique : tous les parenthésages sont testés jusqu'à ce que celui qui correspond à la forme des types soit trouvé.

Nous proposons d'utiliser ce parallélisme entre analyse syntaxique et apprentissage pour répondre à la problématique de l'apprentissage avec une grammaire initiale. En effet, il est nécessaire pour tirer parti des données de la grammaire initiale (types des mots connus) que le processus de construction de l'ensemble de toutes les structures possibles sur une phrase parte des feuilles (les mots) et non de la racine : sinon, on doit comme dans le cas général d'apprentissage (sans grammaire initiale) calculer d'abord toutes les structures, avant de pouvoir éliminer celles qui sont en contradiction avec les données (cas de l'algorithme naïf ci-dessus). À l'inverse, procéder à partir des mots vers la racine permet de tenir compte dès que possible des contraintes imposées par les types des mots connus, et donc ne pas calculer inutilement les structures en contradiction avec ces données. Selon la proportion de mots inconnus dans la phrase et leur répartition, cela permet de limiter l'explosion combinatoire dûe aux différentes structures possibles.

Cette technique est beaucoup plus proche des méthodes d'analyse syntaxique que de l'apprentissage de type RG. L'algorithme proposé ci-dessous utilise une méthode de parsing incrémental de type CYK, de manière à guider la construction de l'arbre de dérivation par les types fournis dans la grammaire initiale. Ce genre de méthode était déjà utilisée pour l'apprentisage par Dudau-Sofronie et Tellier, mais dans un contexte différent (apprentissage de grammaires à partir d'informations sémantiques) [DUD 04b] [DUD 04a].

4.2. *Algorithme*

L'algorithme RGPL (Rigid Grammars Partial Learning) prend en entrée une phrase $w_1, .., w_n$ et une grammaire initiale G_0, composée de règles associant un type à un mot, de la forme $w \mapsto t$. Il renvoie l'ensemble des grammaires rigides solutions, c'est-à-dire celles contenant les règles de la grammaire initiale et acceptant cette phrase.

De même que dans l'algorithme d'apprentissage à partir de FA-structures, cet algorithme utilise des termes qui peuvent contenir des variables. Ces variables sont pro-

gressivement instanciées selon les contraintes imposées par les règles universelles, en particulier dans le cas où le type est combiné avec un type sans variable. Chaque type "réalisable" par un constituant est accompagné de la substitution qui permet de l'obtenir. Cette substitution porte sur les variables associées aux mots inconnus de ce constituant, afin de vérifier lors de chaque combinaison de types que leurs substitutions sont compatibles (par calcul du MGU).

Plus précisément, cet algorithme utilise une matrice M de taille $n \times n$ (n est le nombre de mots dans la phrase). De manière informelle (et simplifiée), l'objectif est de remplir cette matrice de telle sorte qu'à la fin du processus chaque case $M[i,j]$ contienne l'ensemble des types possibles pour l'intervalle de mots $[w_i, \ldots, w_j]$. Toutes les cases $M[i,i]$ (la diagonale) sont d'abord initialisées soit par le type du mot w_i si celui-ci est connu[8] (appartient au lexique $Lex(G)$), soit par une (nouvelle) variable qui est affectée à ce mot s'il est inconnu. En pratique, chaque type présent dans une case sera accompagné de la substitution qui permet de l'obtenir (car les variables des mots inconnus peuvent se développer de différentes façons, qui doivent toutes être conservées). La partie principale de l'algorithme est réalisée à l'intérieur du parcours "CYK", qui consiste à effectuer le parcours de tous les intervalles du plus petit au plus grand (boucles sur deb et fin) et de toutes les combinaisons binaires de sous-intervalles (boucle sur k). Chaque paire de couples type/substitution des deux sous-intervalles est ensuite testée : si les substitutions sont compatibles (existence d'un unifieur le plus général) et que la forme des types correspond à l'application d'une des deux règles universelles (le fait que le "motif" de la partie gauche de la règle corresponde au couple de types est testé encore par unification), alors un nouveau couple type/substitution est ajouté pour l'intervalle en cours d'analyse. À la fin de l'analyse, les grammaires solutions sont obtenues par application des substitutions sur le lexique (contenant des variables pour les mots inconnus). Les substitutions valides sont celles qui permettent d'obtenir le type spécial S comme type de la phrase complète, c'est-à-dire pour l'intervalle $[w_1, \ldots, w_n]$.

Il est important de noter que malgré son fonctionnement "de type CYK" la complexité de cet algorithme n'est pas polynomiale en général. En effet, le nombre de couples (T, σ) dans chaque case de la matrice M peut croître de manière exponentielle. C'est notamment le cas lorsque tous les mots sont inconnus (la grammaire initiale est vide) : on se trouve alors dans le cas précédent d'apprentissage à partir de phrases plates, et l'algorithme doit calculer l'ensemble de toutes les FA-structures possibles.

Cet algorithme ne décrit le fonctionnement que pour une phrase, c'est-à-dire qu'il correspond seulement à la première phase de l'algorithme RG (construction de la grammaire générale). Le processus d'apprentissage sur un ensemble de phrases consiste

8. Comme la contrainte de rigidité n'est nécessaire que pour la partie inconnue de la grammaire, il est parfaitement envisageable que la grammaire initiale ne soit pas rigide, du moment que tous les mots qu'elle contient soient totalement défi nis. Cependant nous nous contentons ici de présenter la version "grammaire rigide seulement", parce que cette extension est d'une utilité limitée et rend l'algorithme légèrement plus complexe.

à appliquer RGPL sur chaque phrase puis calculer le MGU (s'il existe) sur chaque ensemble de solutions, comme dans l'algorithme de Kanazawa.

RGPL(G_0, $[w_1, w_2, .., w_n]$)
  $Lex \leftarrow \{(W,T) \mid (W \mapsto T) \in G_0\}$ % Initialisation
  créer une matrice vide $M[1..n, 1..n]$
  pour $i \leftarrow 1$ à n faire
    si $\exists\, T$ tel que $(w_i, T) \in Lex$ alors
      $M[i,i] \leftarrow \{(T, Id)\}$
    sinon
      créer une nouvelle variable V
      $Lex \leftarrow Lex \cup \{(w_i, V)\}$
      $M[i,i] \leftarrow \{(V, Id)\}$
    fin si
  fin pour
  pour $fin \leftarrow 2$ à n faire % Processus de dérivation/apprentissage partiel
    pour $deb \leftarrow fin - 1$ à 1 faire
      pour $k \leftarrow deb$ à $fin - 1$ faire
        pour chaque $(T_l, \sigma_l) \in M[deb, k]$ faire
          pour chaque $(T_r, \sigma_r) \in M[k+1, fin]$ faire
            si $\exists\, \sigma_u = mgu(\sigma_l, \sigma_r)$ alors
              créer deux nouvelles variables A, B
              si $\exists\, \sigma_{FA} = mgu(\{\{\sigma_u(T_l), A/B\},\ \{\sigma_u(T_r), B\}\})$ alors
                $M[deb, fin] \leftarrow M[deb, fin] \cup \{(\sigma_{FA}(A), \sigma_{FA} \circ \sigma_u \circ \sigma_l)\}$
              fin si
              créer deux nouvelles variables A', B'
              si $\exists\, \sigma_{BA} = mgu(\{\{\sigma_u(T_l), B'\},\ \{\sigma_u(T_r), B' \setminus A'\}\})$ alors
                $M[deb, fin] \leftarrow M[deb, fin] \cup \{(\sigma_{BA}(A'), \sigma_{BA} \circ \sigma_u \circ \sigma_l)\}$
              fin si
            fin si
          fin pour
        fin pour
      fin pour
    fin pour
  fin pour
  $Res \leftarrow \emptyset$ % Application des substitutions compatibles
  pour chaque $(T, \sigma) \in M[1, n]$ faire
    si $\exists\, \tau$ tel que $\tau(T) = S$ alors
      $Res \leftarrow Res \cup \{(\tau \circ \sigma)(Lex)\}$
    fin si
  fin pour
  renvoyer Res
Fin RGPL

Remarque : Id désigne la substitution identité, et $(\sigma_u \circ \sigma_l) = (\sigma_u \circ \sigma_r)$ car σ_u est défini comme le MGU de σ_l et σ_r.

4.3. *Exemple*

Soit G_0 la grammaire initiale définie par le lexique suivant[9]

$\{un : SN/N,\ homme : N,\ poisson : N,\ nage : SN\backslash S,\ vite : (SN\backslash S)\backslash(SN\backslash S)\}$.

Soit "$un\ homme\ court$" la phrase donnée comme entrée à l'algorithme, où "$court$" est un mot inconnu.

1) *Initialisation :*
$M[1,1] \leftarrow (SN/N, \emptyset)$
$M[2,2] \leftarrow (N, \emptyset)$
$M[3,3] \leftarrow (x_1, \emptyset)$ et $Lex \leftarrow Lex \cup \{(court, x_1)\}$

2) *Dérivation :*

$i=2, j=1, k=1:$	$M[1,2] \leftarrow (SN, \emptyset)$	(FA)
$i=3, j=2, k=2:$	$M[2,3] \leftarrow (x_2, \{x_1 \mapsto N\backslash x_2\})$	(BA)
$i=3, j=1, k=1:$	$M[1,3] \leftarrow (SN, \{x_1 \mapsto N\backslash N\})$	(FA)
$i=3, j=1, k=1:$	$M[1,3] \leftarrow (x_3, \{x_1 \mapsto N\backslash((SN/N)\backslash x_3)\})$	(BA)
$i=3, j=1, k=2:$	$M[1,3] \leftarrow (x_4, \{x_1 \mapsto SN\backslash x_4)\})$	(BA)

3) *Substitutions compatibles avec S :*

$\tau(SN) = S$?	impossible
$\tau(x_3) = S$?	$\{x_1 \mapsto N\backslash((SN/N)\backslash S)\}$
$\tau(x_4) = S$?	$\{x_1 \mapsto SN\backslash S\}$

Après cet exemple, il y a deux grammaires dans l'ensemble des solutions : l'une définit "$court$" par le type $N\backslash((SN/N)\backslash S)$, l'autre par le type $SN\backslash S$. Si l'exemple "$un\ homme\ court\ vite$" apparaît plus tard dans la séquence d'exemples, la première solution sera éliminée, car il n'existe pas de substitution sur les règles universelles permettant de combiner $N\backslash((SN/N)\backslash S)$ avec le type de l'adverbe $vite$, $(SN\backslash S)\backslash(SN\backslash S)$.

4.4. *Discussion*

4.4.1. *Contrainte de rigidité et grammaire minimale*

L'algorithme RGPL présenté ci-dessus apprend uniquement des grammaires rigides. Plus exactement, les mots définis dans la grammaire initiale peuvent avoir plusieurs catégories, mais l'algorithme unifie tous les types d'un même mot inconnu. Il ne serait bien sûr pas difficile de gérer des grammaires k-valuées, mais cela serait contraire à l'objectif puisque l'algorithme deviendrait rapidement inexploitable : il serait en effet nécessaire de recourir à la méthode proposée par Kanazawa, qui consiste à calculer toutes les possibilités d'unifier ou non chaque couple de types d'un même mot

9. On peut noter dans cette grammaire que le type des noms N est un argument du type du déterminant SN/N, et non l'inverse : il s'agit de la notation classique dans les grammaires catégorielles.

(voir partie 3.3). De la même façon, nous n'apportons pas ici de solution au problème du calcul d'une grammaire minimale.

On peut par contre envisager pour pallier ces problèmes que des classes (syntaxiques) de mots soit définies dans la grammaire initiale : chaque mot inconnu devrait alors appartenir à l'une des classes prédéfinies, ce qui limite fortement les combinaisons de types. Le regroupemement par classes est fréquemment utilisé (par exemple dans l'étiqueteur de Brill [BRI 93]), mais suppose que les classes prédéfinies couvrent l'ensemble des combinaisons syntaxiques suceptibles d'apparaître dans les exemples, et ne causent pas de surgénération. Même si elle présente un indéniable intérêt pratique, cette solution est contestable sur le plan théorique puisqu'alors l'algorithme ne fait plus vraiment d'inférence grammaticale au sens strict du modèle de Gold.

4.4.2. *La grammaire initiale*

L'intérêt de la méthode d'apprentissage que nous proposons repose entièrement sur l'existence d'une grammaire initiale. Celle-ci doit être suffisamment complète pour qu'un nombre significatif de mots soient connus dans les phrases de la séquence à apprendre. Dans ce cadre, on peut tirer profit de la "loi de Zipf", utilisée notamment par l'étiqueteur de Brill [BRI 93]. Celle-ci garantit que, sur l'ensemble des mots d'un texte, une faible proportion des mots suffit à représenter une grande partie du texte (en nombre d'occurences). Or précisément ce sont les mots les plus fréquents d'un langage qui sont le plus facile à répertorier et définir (mots grammaticaux tels que déterminants, pronoms, prépositions, conjonctions, etc.), soit manuellement soit par conversion de dictionnaires existants dans d'autres formalismes. Il est important de noter que ces mots grammaticaux forment des classes fermées, c'est-à-dire qu'il est relativement facile d'en faire une liste exhaustive, contrairement aux mots tels que noms, verbes ou adjectifs (classes ouvertes). Nous proposons dans la partie 5 quelques tests qui mettent en évidence l'intérêt de la loi de Zipf pour l'apprentissage partiel.

4.4.3. *Autres formalismes*

L'étude réalisée dans cet article porte uniquement sur les grammaires AB, la forme la plus simple de grammaires catégorielles. Ce formalisme est plutôt pauvre en termes de représentation des phénomènes linguistiques, c'est pourquoi il serait souhaitable d'étendre la méthode à des cadres plus adaptés au langage naturel. Il existe différents travaux qui tendent à montrer que le type d'apprentissage proposé par Kanazawa est généralisable, notamment à d'autres formes de grammaires catégorielles telles les grammaires de Lambek[10] et minimalistes [BON 01]. D'autres formalismes, comme les grammaires de liens [SLE 91], sont aussi suceptibles de permettre ce type d'apprentissage, comme nous le montrons dans la partie suivante. D'une manière générale, tous les formalismes totalement lexicalisés bénéficiant de propriétés d'apprenabilité

10. L'adaptation aux grammaires de Lambek pose cependant d'autres problèmes de complexité, puisque le problème de la simple appartenance d'une phrase au langage est NP-complet dans ce formalisme [PEN 03].

dans le modèle de Gold et qui sont exprimables à l'aide de règles de réécriture à base d'unification sont suceptibles d'être utilisables dans le cadre de l'apprentissage partiel.

5. Expérimentation avec les grammaires de liens

Les premiers tests de l'apprentissage partiel présentés dans cette partie utilisent le lexique des grammaires de liens pour l'anglais. Cette grammaire a été construite par Sleator et Temperley [TEM 91] et offre une description assez précise et complète (au niveau grammatical) de l'anglais. L'apprenabilité des grammaires de liens k-valuées a été démontrée par Béchet dans [BÉC 03]. Dans notre cas, les grammaires de liens ont surtout l'avantage d'être très proches des grammaires catégorielles, ce qui rend l'adaptation de l'algorithme relativement aisée. Il est cependant nécessaire d'utiliser un formalisme intermédiaire spécifique à cette tâche : celui-ci est décrit dans la partie 5.2, ainsi que les limites qui en résultent.

5.1. *Grammaires de liens*

Les grammaires de liens (*link grammars*) sont définies par Sleator et Temperley dans [SLE 91]. Il s'agit d'un formalisme relativement simple, qui permet néanmoins de représenter de manière précise un langage naturel. Les auteurs ont ainsi modélisé la langue anglaise dans ce formalisme, de manière à gérer la plupart des aspects grammaticaux de cette langue. Ils ont aussi réalisé un analyseur syntaxique basé sur ce modèle [TEM 91]. Dans notre perspective de l'apprentissage basé sur les grammaires de type grammaires catégorielles, le travail de Sleator et Temperley est particulièrement intéressant pour les raisons suivantes, aussi bien théoriques que pratiques :

– De manière générale, la simplicité du modèle des grammaires de liens offre des possibilités très intéressantes d'adaptation à différents objectifs, tout en ayant une capacité de représentation du langage assez robuste.

– Il existe un rapport assez étroit entre les grammaires de liens et les grammaires catégorielles classiques : ces deux formalismes sont totalement lexicalisés, et basés sur une notion de dépendance.

– Il est possible d'exprimer le modèle des grammaires de liens à l'aide d'un petit ensemble de règles basées sur l'unification. Ce point est essentiel pour notre utilisation de ce modèle dans le cadre de l'apprentissage partiel.

Le principe général des grammaires de liens est qu'une phrase est correcte s'il est possible d'établir des liens entre les mots, selon les contraintes imposées par les définitions des mots dans le lexique. Ces liens représentent les relations syntaxiques entre les mots. Une grammaire de liens est définie par un vocabulaire, un ensemble de connecteurs et une relation qui met en correspondance chaque mot avec un ensemble de types. Un type est un couple de listes ordonnées de connecteurs. Une phrase donnée appartient au langage décrit par la grammaire de liens s'il est possible de dessiner tous ses liens au dessus des mots de manière à satisfaire les critères suivants :

Planarité : Les liens (dessinés au dessus des mots) ne se croisent pas[11].

Connectivité : Il existe un chemin (de liens) entre tout couple de mots de la phrase.

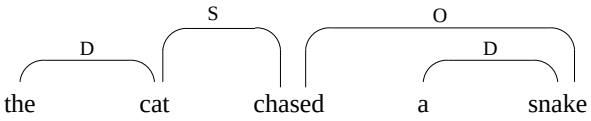
Ordre : Pour chaque mot w de la phrase, il existe un type qui lui est associé dans le lexique tel que ce type est saturé. Un type $d = \texttt{d([}L_1\texttt{,..,}L_l\texttt{], [}R_1\texttt{,..,}R_r\texttt{])}$ est saturé si pour tout connecteur L_i (resp. R_i) il existe un lien étiqueté par le nom du connecteur venant de gauche (resp. de droite) vers w, et pour tout couple de connecteurs L_i, L_j (resp. R_i, R_j) si $i < j$ alors le mot connecté par L_i (resp. R_i) est plus proche de w que celui connecté par L_j (resp. R_j).[12]

Exclusion : Deux liens ne peuvent pas relier le même couple de mots.

Ces critères sont appelés les méta-règles des grammaires de liens. Les auteurs montrent dans [SLE 91] que la classe de langages des grammaires de liens est équivalente à celle des grammaires hors-contexte.

Exemple 4. *Avec le lexique suivant, la phrase ci-dessous est correcte :*

a,the $\mapsto$ `d([],[D])`
cat, snake $\mapsto$ `d([D],[S])`, `d([O,D],[])`
chased $\mapsto$ `d([S],[O])`



5.2. *Grammaires catégorielles de liens*

Les grammaires de liens ont des caractéristiques communes avec les grammaires catégorielles : les deux formalismes sont totalement lexicalisés, et partagent la propriété de relier les mots ou les constituants à l'aide d'une relation binaire de dépendance. Cependant cette relation de dépendance est différente dans les deux systèmes. Dans les grammaires de liens, les dépendances ne sont pas orientées et ne s'appliquent qu'aux mots, contrairement aux grammaire catégorielles dans lesquelles les dépendances sont orientées et s'appliquent aux constituants de la phrase.

Afin de souligner le rapport entre les grammaires de liens et les grammaires catégorielles, nous utiliserons un formalisme intermédiaire nommé *grammaires catégorielles de liens* (GCL), dans lequel les dépendances sont orientées et s'appliquent aux

11. Formellement : s'il existe deux liens l et l' entre deux couples de mots (w_i, w_j) et $(w_{i'}, w_{j'})$ (où w_k est le k^{eme} mot de la phrase), alors l'une des inégalités suivantes est vérifiée : $i < j \leq i' < j'$ ou $i' < j' \leq i < j$.
12. *Remarque :* dans la définition originale la liste de connecteurs à droite est notée $\texttt{[}R_r\texttt{,...,}R_1\texttt{]}$. Nous choisissons ici d'inverser l'ordre pour maintenir la cohérence avec la notation `cons(c,L)/nil` présentée dans la partie 5.2.

constituants. Ce formalisme permettra d'adapter facilement les méthodes d'apprentissage issues des grammaires catégorielles aux grammaires de liens. En fait le formalisme des GCL est simplement une interprétation différente des règles classiques des grammaires de liens. Le formalisme proposé est équivalent en termes de langages aux grammaires de liens, à l'exception du fait que les réseaux de liens ne doivent pas contenir de cycles (voir [MOR 04] pour plus de détails sur ce point).

La définition d'une grammaire en GCL est identique à celle des grammaires de liens, mais les règles de dérivation sont désormais exprimées comme la réécriture d'une séquence de termes en une autre, comme pour les grammaires catégorielles. C'est pourquoi les liste de connecteurs sont notées comme des termes (non associatifs) utilisant les opérateurs classiques de listes `cons` et `nil` :

$$\texttt{cons(}c_1\texttt{,cons(}c_2\texttt{,..cons(}c_n\texttt{,nil)..))}$$

est équivalent à $\texttt{[}c_1\texttt{,}c_2\texttt{,..,}c_n\texttt{]}$. Les règles de dérivation des GCL sont décrites ci-dessous, avec `c` un connecteur et `L` et `R` des listes (éventuellement vides) de connecteurs :

$$\texttt{d(L, cons(c, R)), d(cons(c, nil), nil)} \rightarrow \texttt{d(L,R)}$$
$$\texttt{d(nil, cons(c, nil)), d(cons(c, L), R)} \rightarrow \texttt{d(L,R)}$$

La première règle (qui s'écrit "`d(L, [c|R]), d([c], []) → d(L,R)`" avec la notation des listes Prolog, plus lisible) signifie qu'un constituant dont le premier connecteur droit est `c` peut consommer son constituant le plus proche à droite si (et seulement si) celui-ci a pour unique connecteur `c` à gauche, et aucun connecteur à droite. La seconde règle, qui sert à consommer le constituant de gauche, est symétrique. De proche en proche, l'application de ces règles garantit que les méta-règles sont respectées, générant ainsi le même langage que la grammaire de liens "classique" correspondante.

Soit $\Rightarrow$ la relation définie par $\alpha\, t_1\, t_2\, \beta \Rightarrow \alpha\, t_3\, \beta$ si et seulement si $t_1\, t_2 \rightarrow t_3$. On définit la relation $\Rightarrow^*$ comme la fermeture transitive et réflexive de $\Rightarrow$. Une phrase $w_1\, w_2\, \ldots\, w_n$ est correcte pour une grammaire GCL s'il existe une séquence de types $t_1, t_2, \ldots, t_n$ telle que pour tout i t_i est l'un des types de w_i, et $t_1\, t_2\, \ldots\, t_n \Rightarrow^* d(nil, nil)$.

Exemple 5. *La phrase de l'exemple 4 admet la dérivation suivante en GCL (avec le même lexique) :*

	the	*cat*	*chased*	*a*	*snake*
	`d([],[D])`,	`d([D],[S])`,	`d([S],[O])`,	`d([],[D])`,	`d([D,O],[])`
$\Rightarrow$		`d([],[S])`,	`d([S],[O])`,	`d([],[D])`,	`d([D,O],[])`
$\Rightarrow$		`d([],[S])`,	`d([S],[O])`,		`d([O],[])`
$\Rightarrow$		`d([],[S])`,		`d([S],[])`	
$\Rightarrow$			`d([],[])`		

5.3. *Prototype*

Le fait que les auteurs Sleator et Temperley aient construit un lexique de la langue anglaise dans le formalisme des grammaires de liens constitue un avantage pratique non négligeable. Ce lexique contient approximativement 59000 mots, répartis dans 1350 classes syntaxiques, chaque classe correspondant à un ensemble de types. La grammaire gère une grande partie des phénomènes linguistiques de l'anglais, ce qui selon les auteurs *"indique que cette approche peut avoir des utilisations pratiques et être pertinente au niveau linguistique"*. De plus, l'analyseur syntaxique fourni par les auteurs inclut un fichier d'exemples de 928 phrases, étiquetées comme correctes ou incorrectes.

Nous décrivons brièvement ci-dessous comment le lexique d'origine a été converti dans le formalisme des grammaires catégorielles de liens. Cette conversion prend en compte certaines des "caractéristiques avancées" qui ont été ajoutées par les auteurs au système de base des grammaires de liens.

5.3.1. *Multi-connecteurs et format du lexique*

Les itérations sont représentées à l'aide de *multi-connecteurs* : tout connecteur `C` dans une liste de connecteurs peut être précédé du symbole `@`, ce qui signifie que ce connecteur `@C` peut être relié à plusieurs liens de type `C`. Par exemple, les noms peuvent avoir le type `d([@A,D],[S])`, de manière à autoriser qu'un nombre quelconque d'adjectifs précède le nom. Cette possibilité est maintenue dans le formalisme des GCL, en remplaçant les "règles de base" par les suivantes (dans lesquelles `[@]` signifie que le symbole `@` est optionnel) :

```
    d(L, cons([@]c, R)) , d(cons([@]c, nil), nil) → d(L,R)
    d(nil, cons([@]c, nil)) , d(cons([@]c, L), R) → d(L,R)
d(L, cons(@c, R)) , d(cons([@]c, nil), nil) → d(L,cons(@c, R))
d(nil, cons([@]c, nil)) , d(cons(@c, L), R) → d(cons(@c, L),R)
```

Les types sont définis dans le lexique d'origine comme des formules utilisant des opérateurs et/ou, dans lesquelles les connecteurs à gauche sont notés `C-` et les connecteurs à droite sont notés `C+`. Un opérateur `?` est aussi utilisé pour les sous-formules optionnelles. La conversion remplace chaque formule par la liste de types équivalentes, ce qui correspond pratiquement à en calculer la forme normale disjonctive.

5.3.2. *Subscripts*

Les *subscripts* sont utilisés pour spécialiser les connecteurs, d'une manière similaire aux structures de traits dans les grammaires d'unification. Ils sont utilisés pour rendre la grammaire plus facile à comprendre et éventuellement à modifier. Par exemple, les subscripts `s` et `p` sont utilisés avec les noms (ainsi que les pronoms, déterminants, etc.) pour distinguer ceux qui sont au singulier de ceux qui sont au pluriel. Pour maintenir la simplicité des règles de réduction, les types contenant des subscripts

sont convertis en types sans subscripts. Ceci est réalisé à l'aide d'un programme qui classifie tous les connecteurs existants (avec leurs subscripts) de telle sorte que tous les connecteurs d'une même classe ne puisse être reliés qu'avec le même ensemble de connecteurs. Afin de minimiser le nombre de classes, deux classes sont fusionnées chaque fois que cela est possible. A la fin du processus un nouveau type est créé pour chaque classe : le lexique ainsi converti est équivalent au lexique d'origine.

5.3.3. *L'analyseur syntaxique de GCL*

La correction de la conversion a été testée en comparant les résultats de l'analyse syntaxique des phrases du fichier d'exemples avec ceux obtenus par l'analyseur fourni par les auteurs. L'analyseur de GCL est un analyseur standard de type CYK, non optimisé, qui applique simplement les règles de réduction binaires définies plus haut. Cet analyseur ne gère pas certaines des fonctionnalités avancées de l'analyseur d'origine (conjonctions, règles de post-traitement, système de coût et mots inconnus), et l'ensemble des exemples a été modifié en conséquence : après la suppression des phrases nécessitant ces fonctionnalités pour être traitées correctement, on obtient un échantillon de 771 phrases parmi lesquelles 221 sont incorrectes. Une autre simplification importante concerne le problème de la conversion des cycles[13]. Les expérimentations présentées dans cet article sont réalisées avec une version "sans cycles" (simplifiée) de la grammaire : celle-ci n'est donc pas capable de reconnaître certaines erreurs syntaxiques dans les phrases. Ainsi 38 phrases incorrectes sont considérées comme correctes par l'analyseur syntaxique (4,9% des phrases de l'échantillon sont donc mal analysées).

Exemple 6. *Voici certaines phrases contenues dans le fi chier d'exemples, les phrases incorrectes étant précédées d'une astérisque :*

What did John say he thought you should do
**What did John say did he think you should do*
To pretend that our program is usable in its current form would be silly
That our program will be immediately accepted is hardly likely
**Is that our program will be accepted likely*
The man there was an attempt to kill died

5.4. *Expérimentations et résultats*

L'objectif des expérimentations présentées ci-dessous est de tester la faisabilité en pratique de l'apprentissage partiel avec les données issues du lexique anglais des grammaires de liens. Le prototype utilisé est codé en SWI Prolog. Il est important de noter que la seule partie du problème observée dans ces tests est la possibilité de réaliser l'étape de construction des grammaires générales par apprentissage partiel

13. Ce problème (qui sort du cadre de cet article) est détaillé dans [MOR 04], où différentes solutions sont envisagées.

en un temps raisonnable : aucune phase d'unification ni de calcul de la grammaire minimale n'est réalisée (voir discussion dans la partie 4.4.1).

5.4.1. *Le prototype d'apprentissage partiel des GCL*

Le prototype d'apprentissage partiel est basé sur l'analyseur syntaxique de grammaires catégorielles de liens présenté en 5.3, et utilise l'algorithme décrit en 4.2. Le langage Prolog est clairement bien adapté pour manipuler des termes contenant des variables au cours du processus : en pratique les substitutions ne sont pas stockées dans la structure de données comme indiqué dans l'algorithme, mais simplement obtenues à l'aide du mécanisme d'unification de Prolog. L'analyse demeure néanmoins déterministe : les types sont copiés chaque fois que c'est nécessaire de manière à maintenir des listes de types dans la matrice, et ainsi éviter le *backtracking* sur la matrice.

En dehors de quelques optimisations non essentielles, le programme a aussi une caractéristique importante qui permet d'éviter une partie conséquente de l'explosion combinatoire. Il s'agit de la factorisation des types "structurellement équivalents", sachant que deux types t et t' sont *structurellement équivalents* s'il existe deux substitutions σ_1 et σ_2 telles que $\sigma_1(t) = t'$ et $\sigma_2(t') = t$ (en d'autres termes les deux types sont identiques modulo un renommage de variables). Dans la mesure où il est fréquent que deux types structurellement équivalents appartiennent à la même case de la matrice, cette optimisation diminue de manière significative le temps d'exécution et l'espace mémoire utilisé. Cependant ce mécanisme nécessite que la composition des types ne soit plus calculée à chaque étape au cours de l'analyse, sinon il faudrait redévelopper tous les types dès qu'on passe au niveau suivant. C'est pourquoi seule la substitution utilisée pour transformer un type à un niveau est conservée. Le programme doit donc réaliser une seconde phase qui calcule les compositions globales des substitutions, après la phase d'analyse incrémentale.

Les tests sont faits à partir du fichier d'exemples fourni avec l'analyseur syntaxique des grammaires de liens [TEM 91]. Seule une partie des phrases correctes de cet ensemble d'exemples a été conservée. La segmentation en mots des phrases est réalisée (et vérifiée manuellement) avant le processus d'apprentissage partiel. Les tests consistent à supprimer une partie des mots du lexique, de manière à ce que ces mots soient considérés comme inconnus. Selon le nombre de mots retirés du lexique, et surtout selon le nombre d'occurences de ces mots dans l'échantillon, la limite d'application de l'algorithme (dûe à l'explosion combinatoire) sera rencontrée plus ou moins vite. C'est pourquoi on testera différents taux de mots inconnus jusqu'à s'approcher de cette limite, dans les deux cas suivants : d'abord en supprimant aléatoirement certains mots du lexique, cette mesure servant alors de référence, puis en éliminant prioritairement du lexique les mots les moins fréquents de l'échantillon, de manière à simuler l'utilisation d'un lexique initial constitué de mots fréquents.

5.4.2. *Résultats*

L'échantillon utilisé contient 537 phrases et 4765 mots (taille de l'échantillon), dont 1064 mots différents (taille du lexique). Les mots les plus fréquents sont *the*

(270), *I* (155), *is* (122), *to* (121). Dans ces tests nous comparons les temps de calcul de l'apprentissage partiel pour différents taux de mots inconnus (dans le lexique et dans l'échantillon).

Dans les résultats ci-dessous "MI" désigne les mots inconnus : la première colonne (lexique) indique le nombre de mots inconnus parmi l'ensemble des mots, tandis que la seconde colonne (échantillon) indique le nombre d'occurences de ces mots inconnus dans les phrases. Dans ce premier test les mots supprimés du lexique sont choisis aléatoirement. Les résultats sont présentés dans le tableau 1 : on peut voir que le taux de mots inconnus dans l'échantillon est proche de celui dans le lexique.

MI (lexique)	MI (échantillon)	Temps total	MI par phrase (min ; moy ; max)	temps par phrase (min ; moy ; max) (sec)
0 (0%)	0 (0%)	27 min	0 ; 0 ; 0	0,1 ; 3,1 ; 14,7
106 (10%)	407 (8,5%)	75 min	0 ; 0,7 ; 4	0,1 ; 8,3 ; 333
211 (20%)	870 (18,2%)	3,5 h	0 ; 1,5 ; 6	0,1 ; 24 ; 1985
264 (25%)	1106 (23,2%)	3,8 h	0 ; 1,8 ; 7	0,1 ; 24,9 ; 1115
317 (30%)	1481 (31,1%)	Échec (mémoire insuffisante)		

Tableau 1. *Suppression aléatoire de mots du lexique.*

Afin de simuler l'idée qu'il est plus judicieux de construire la grammaire initiale en utilisant un petit ensemble de mots (très) fréquents, seuls les mots les moins fréquents sont supprimés du lexique dans ce second test. Grâce à la loi de Zipf, il est ainsi possible de supprimer une grande partie des mots du lexique sans avoir trop de mots inconnus dans l'échantillon, comme le montre le tableau 2.

MI (lexique)	MI (échantillon)	Temps total	MI par phrase (min ; moy ; max)	temps par phrase (min ; moy ; max) (sec)
0 (0%)	0 (0%)	27 min	0 ; 0 ; 0	0,1 ; 3,1 ; 14,7
211 (20%)	211 (4,4%)	45 min	0 ; 0,3 ; 4	0,1 ; 5,0 ; 173
422 (40%)	422 (8,8%)	78 min	0 ; 0,6 ; 5	0,1 ; 8,7 ; 620
634 (60%)	727 (15,2%)	2,6 h	0 ; 1,1 ; 7	0,1 ; 17,3 ; 1844
845 (80%)	1302 (27,3%)	5,1 h	0 ; 2,0 ; 9	0,1 ; 34,1 ; 1893

Tableau 2. *Suppression du lexique des mots les moins fréquents.*

Ces tests montrent que le processus d'apprentissage partiel peut fonctionner en un temps acceptable (et sans surcharger l'espace mémoire) avec un taux de mots inconnus dans l'échantillon allant jusqu'à 25 à 30%. On peut constater qu'il suffit qu'environ 20% des mots du lexique soient définis pour obtenir un tel taux, grâce à la loi de Zipf.

Bien entendu, les données sur lesquelles sont réalisées ces tests ne sont pas suffisamment représentatives pour garantir que l'algorithme serait en mesure de fonctionner dans toute situation où le taux de mots inconnus serait celui-ci. Mais le lexique anglais des grammaires de liens est une bonne approximation d'un langage naturel, et

n'était pas conçu pour ce type d'application. Par conséquent ces tests semblent montrer que la méthode de l'apprentissage partiel pourrait être une approche réaliste pour appliquer l'apprentissage symbolique aux langues naturelles.

6. Conclusion et perspectives

Dans le domaine de l'inférence grammaticale, l'apprenabilité des classes de langages est souvent étudiée indépendamment d'une éventuelle mise en pratique de cet apprentissage. Du strict point de vue de l'apprenabilité dans le modèle de Gold, aucune distinction n'est nécessaire entre les cas d'apprentissage de grammaires rigides à partir de FA-structures et de grammaires k-valuées à partir de phrases plates, même si le premier est peu utile et le second pratiquement irréalisable.

Dans la perspective d'applications "réelles", nous avons proposé une adaptation des algorithmes d'apprentissage de grammaires catégorielles. L'approche étudiée est moins contraignante sur la forme des entrées de l'algorithme, tout en restant relativement réaliste du point de vue de l'efficacité. Elle est basée sur la lexicalisation totale et l'unification, ce qui en fait une méthode potentiellement utilisable avec de nombreux formalismes. Néanmoins il reste plusieurs questions à résoudre avant de pouvoir appliquer des algorithmes d'apprentissage symbolique de règles syntaxiques aux langues naturelles : comme on l'a vu, le passage aux grammaires k-valuées, indispensable pour garantir une expressivité suffisante des classes de langages concernées, reste une source d'inefficacité des algorithmes. L'utilisation de classes de mots prédéfinis pourrait être une piste intéressante sur ce point, mais à condition d'y trouver un intérêt spécifique (car cela ne serait plus de l'inférence grammaticale "pure", mais ne serait pas nécessairement plus utile que les étiqueteurs syntaxiques existants).

Sur un plan pratique, on peut s'interroger sur ce qu'un algorithme d'apprentissage doit renvoyer comme résultat : dans l'idéal ce devrait être une unique grammaire, mais le calcul de la grammaire minimale parmi un ensemble de grammaires solutions est définitivement trop complexe, quand il n'est pas impossible. Même si l'on tolère que la réponse de l'algorithme soit un ensemble de grammaires, cet ensemble risque souvent d'être beaucoup trop grand pour être calculable. On serait alors tenté d'utiliser des représentations plus flexibles pour les types, comme dans la version originale des grammaires de liens où les types sont des formules contenant des opérateurs logiques (*et*, *ou*, etc.) ce qui permet de factoriser les similarités (donc représenter plusieurs grammaires en une seule). Mais l'apprentissage symbolique a besoin de pouvoir reconnaître lorsque deux types sont identiques, ce qui suppose que leur représentation soit normalisée.

Globalement, il semble donc très difficile de respecter toutes les contraintes imposées par le modèle de Gold dans le cadre d'un apprentissage automatique réel (sur de grandes quantités de données) appliqué aux langues naturelles. Cependant il ne faut sans doute pas en déduire trop vite que les obstacles sont insurmontables : malgré le pessimisme initial qui a longtemps pesé sur ce modèle, il s'est avéré en mesure de re-

présenter des distinctions non évidentes entre classes apprenables et non apprenables. Mais même s'il est nécessaire de sortir du cadre strict du modèle de Gold pour trouver des applications aux algorithmes d'apprentissage qui en découlent, par exemple en utilisant des types fixés dans des classes de mots prédéfinis, l'apprentissage symbolique a principalement un avantage : la précision. En effet, cette forme d'apprentissage garantit une précision maximale qui peut être un avantage important dans certaines tâches, même s'il est probable que l'efficacité en restera le principal défaut.

7. Bibliographie

[ANG 80a] ANGLUIN D., « Finding patterns common to a set of strings. », *Journal of Computer and System Sciences*, vol. 21, 1980, p. 46–62.

[ANG 80b] ANGLUIN D., « Inductive inference of formal languages from positive data », *Information and Control*, vol. 48, 1980, p. 117–135.

[ANG 83] ANGLUIN D., SMITH C. H., « Inductive Inference : Theory and Methods », *ACM Computing Surveys*, vol. 15, n° 3, 1983, p. 237–269.

[BAR 53] BAR-HILLEL Y., « A Quasi-Arithmetical Notation for Syntactic Description », *Language*, vol. 29, 1953.

[BAR 60] BAR-HILLEL Y., GAIFMAN C., SHAMIR E., « On Categorial and Phrase Structure Grammars », *Bulletin of the Research Council of Israel*, vol. 9F, 1960.

[BÉC 03] BÉCHET D., « k-Valued Link Grammars are Learnable From Strings », *Proceedings of Formal Grammars 2003*, 2003, p. 9–18.

[BES 03] BESOMBES J., MARION J.-Y., « Apprentissage de langages réguliers d'arbres et applications », *Traitement Automatique des Langues*, vol. 44, n° 1, 2003, p. 121–153, Hermes.

[BON 01] BONATO R., RETORÉ C., « Learning Rigid Lambek Grammars and Minimalist Grammars from Structured Sentences », POPELÍNSKÝ L., NEPIL M., Eds., *Proceedings of the 3d Workshop on Learning Language in Logic*, Strasbourg, France, September 2001, p. 23–34.

[BRI 93] BRILL E., « A Corpus-Based Approach to Language Learning », PhD thesis, Computer and Information Science, University of Pennsylvania, juin 1993.

[BUS 89] BUSZKOWSKI W., PENN G., « Categorial grammars determined from linguistic data by unifi cation », rapport n° TR-89-05, juin 1989, Department of Computer Science, University of Chicago.

[COS 01] COSTA FLORÊNCIO C., « Consistent Identifi cation in the Limit of the Class k-valued is NP-hard », DE GROOTE P., MORRILL G., RETORÉ C., Eds., *Logical Aspects of Computational Linguistics, 4th International Conference, LACL 2001, Le Croisic, France, June 27-29, 2001, Proceedings*, vol. 2099 de *Lecture Notes in Computer Science*, Springer-Verlag, 2001, p. 125–138.

[COS 02] COSTA FLORÊNCIO C., « Consistent Identifi cation in the Limit of Rigid Grammars from Strings is NP-hard », ADRIAANS P., FERNAU H., VAN ZAANEN M., Eds., *Grammatical Inference : Algorithms and Applications 6th International Colloquium : ICGI 2002*, vol. 2484 de *Lecture Notes in Artificial Intelligence*, Springer-Verlag, September 23-25 2002, p. 49–62.

[DUD 04a] DUDAU-SOFRONIE D., TELLIER I., « Un modèle d'acquisition de la syntaxe à l'aide d'informations sémantiques », *actes de la 11ème Conférence TALN, Traitement Automatique du Langage Naturel*, Fès, Maroc, avril 2004, p. 137–146.

[DUD 04b] DUDAU-SOFRONIE D., « Apprentissage de grammaires catégorielles pour simuler l'acquisition du langage naturel à l'aide d'informations sémantiques », PhD thesis, Université Lille 1, avril 2004.

[GOL 67] GOLD E., « Language identification in the limit », *Information and control*, vol. 10, n° 5, 1967, p. 447-474.

[KAN 98] KANAZAWA M., *Learnable classes of categorial grammars*, Cambridge University Press, 1998.

[KAP 91] KAPUR S., « Computational Learning of Languages », PhD thesis, Department of Computer Science, Cornell University, Ithaca, New York, 1991.

[KNI 89] KNIGHT K., « Unification : A Multidisciplinary Survey », *ACM Computing Surveys*, vol. 21, n° 1, 1989, p. 93–124.

[MAR 93] MARCUS G., « Negative evidence in language acquisition », *Cognition*, vol. 46, 1993, p. 53–85.

[MOR 04] MOREAU E., « From link grammars to categorial grammars. », *Proceedings of Categorial Grammars 2004*, Montpellier, France, juin 2004, p. 31–45.

[MOT 91] MOTOKI T., SHINOHARA T., WRIGHT K., « The correct definition of finite elasticity : corrigendum to Identification of unions », *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*, San Mateo, CA, 1991, Morgan Kaufmann, page 375.

[NIC 99] NICOLAS J., « Grammatical inference as unification », rapport rr° 3632, juillet 1999, INRIA, également rapport IRISA PI1265.

[PEN 03] PENTUS M., « Lambek calculus is NP-complete », rapport n° TR-2003005, 2003, CUNY Ph.D. Program in Computer Science.

[PIN 94] PINKER S., *The Language Instinct : How the Mind Creates Language*, HarperCollins, New York, 1994.

[SAK 97] SAKAKIBARA Y., « Recent advances of grammatical inference », *Theoretical Computer Science*, vol. 185, n° 1, 1997, p. 15–45.

[SHI 91] SHINOHARA T., « Inductive Inference of Monotonic Formal Systems From Positive Data », *New Generation Computing*, vol. 8, n° 4, 1991, p. 371–384.

[SLE 91] SLEATOR D. D. K., TEMPERLEY D., « Parsing English with a Link Grammar », rapport n° CMU-CS-TR-91-126, 1991, Carnegie Mellon University, Pittsburgh, PA.

[TEM 91] TEMPERLEY D., SLEATOR D., LAFFERTY J., « Link grammar. http ://hyper.link.cs.cmu.edu/link/ », 1991.

[WRI 89] WRIGHT K., « Identification of unions of languages drawn from an identifiable class », *Proceedings of the Second Annual Workshop on Computational Learning Theory*, Morgan Kaufmann, 1989, p. 328–333.